EECS219 Homework II: MapReduce

Due May 13, 11pm

Spring 2015

1. HW2 Project Assignment

In this assignment, you are to build a program to calculate the probability of sentences from a sample of corpus. From the corpus sample, you can get the probability of a word that exists at position i by using the formula below:

P(i, w) = Num(i, w) / N;

where Num(i , w) is the number of times that w exists at i-th position of a sentence, N is the total number of sentences with at least i words.

The probability of a sentence is the product of probability of all its individual words.

1 2 3 4 5 6 7 [positions]

It is the assignment for EECS 219 [words]

P(s) = P(1 , 'It')P(2, 'is')P(3, 'the')P(4, 'assignment')P(5, 'for')P(6, 'EECS')P(7, 219).

You should implement a MapReduce program that

1. accepts two parameters to execute. One is for file path for the corpus, the other is where to place your result;
2. generates the top three sentences with the largest existence probability in the corpus.

We have prepared a small sample file for your testing in your standalone environment,

https://www.dropbox.com/s/ubrivo8sh6x1z1h/Corpus.txt?dl=0

You can look at it to see the file format. We will use a much bigger file to test the correctness and performance of your program in a distributed cluster.

You may need to run more than one jobs sequentially to solve the problem. Please refer to **JobControl** class and **addDependingJob** method of Job class to achieve your goal. To prevent a float point overflow, you maybe need to multiply each P with a constant. (You should determine the scale by checking the smallest P value for that particular position.)

Amazon EMR Test

- Compile and package your source code to code.jar
- Create a S3 bucket
- Upload code.jar and input file into S3 bucket
- Start a EMR cluster to execute the CorpusCaculator program as a job.

HW Submit Instruction

The following is the requirement on your homework. Points may be taken if not followed.

- You main class should be **CorpusCaculator** for the convenience of grading.
- Please create a report to demonstrate your experience of using EMR
- Please organize your project in the following directory hierarchy:
  ```
  project2-studentID/codebase/code.jar   (Tested on EMP)
                    /codebase/source/*.java
                    /{readme.txt,  report}
  ```
- Compress project2-studentID into a single ZIP file, with the name "project2-studentID.zip"
- Put the corpus sample as "corpus" file in the root of HDFS, and download the script from https://www.dropbox.com/s/q32f4mm9iaqd61t/test.sh?dl=0. Put the script and your zip file under the same directory. Run it to check whether your project can be properly unzipped and tested. (Run the script by using the command line: **./test.sh 'project2-studentID'**)

2. Preliminary

1. **Map Reduce:** Read the paper http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/archive/mapreduce-osdi04.pdf
2. **Google File System:** Read the paper http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/archive/gfs-sosp2003.pdf
3. **HDFS:** Read the document http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf
4. **Word Count Example** http://hadoop.apache.org/docs/r1.0.4/mapred_tutorial.html

3. Local Development Environment Setup

This section helps you setup a local instance of HDFS and MapReduce runtime. For the cluster-based configuration please refer to Hadoop's official website: http://hadoop.apache.org/docs/r1.1.2/cluster_setup.html

Before you follow the following steps, make sure you have installed Java on your Linux or virtual machine.

(1) Download Hadoop

Download Hadoop from Apache Download Mirrors and extract the content of the Hadoop Package to a location of your choice. I am using /opt/hadoop. Make sure to change the owner of all files to the current user. Please use the version 1.0 rather than other versions, otherwise there will be some unexpected issues when integrating it with Hbase. Download it from the link below:

http://apache.tradebit.com/pub/hadoop/common/hadoop-1.1.2/hadoop-1.1.2.tar.gz

```
$ cd /opt/hadoop

$ sudo tar xzf hadoop-1.1.2.tar.gz

$ sudo mv hadoop-1.1.2 hadoop

$ sudo chown -R {username} hadoop
```

Your HADOOP_HOME should be /opt/hadoop/hadoop for running the submission script.

(2) Change .bashrc of login user

Add the following lines to the end of the /home/{username}/.bashrc file of current user. If you use a shell other than bash, you should of course update its appropriate configuration files instead of .bashrc.

```
$ vim .bashrc (Please refer vim manual)
```

Click entry below at the top of .bashrc:

```
# Set Hadoop-related environment variables

export HADOOP_HOME=/opt/hadoop

# Set JAVA_HOME (we will also configure JAVA_HOME directly for
Hadoop later on)

export JAVA_HOME = /usr/lib/jvm/java-6-openjdk

# Add Hadoop bin/ directory to PATH

export PATH=$PATH:$HADOOP_HOME/bin
```

Click esc //it changes vim to command mode

:w   //write the content in buffer to file

:q!   //exit vim

(3) Hadoop Environment Variable Configuration

Open /conf/hadoop-env.sh in the editor of your choice (if you use the installation path in the tutorial, the full path is /opt/hadoop/hadoop/conf/hadoop-env.sh) and set JAVA_HOME

```
export JAVA_HOME = /usr/lib/jvm/java-6-openjdk
```

(4) Create HDFS Folder

We will use the directory /opt/hadoop/tmp in this tutorial. Hadoop's default configuration uses hadoop.tmp.dir as the base temporary directory both for the local file system and HDFS. So don't be surprised if you see Hadoop creating the specified directory automatically on HDFS at some later point.

```
$ sudo mkdir -p /opt/hadoop/tmp

$ sudo chown {username}   /opt/hadoop/tmp

$ sudo chmod  750 /opt/hadoop/tmp
```

(5) Configure HDFS In {HADOOP_HOME}/conf/core-site.xml

```
<property>
<name>hadoop.tmp.dir</name>
<value>/opt/hadoop/tmp</value>
<description>A base for other temporary directories.</description>
</property>

<property>
<name>fs.default.name</name>
<value>hdfs://localhost:54310</value>
<description>The name of the default file system. A URI whose
scheme and authority determine the FileSystem implementation. The
uri's scheme determines the config property (fs.SCHEME.impl) naming
the FileSystem implementation class. The uri's authority is used to
determine the host, port, etc. for a filesystem.</description>
</property>
```

(6) Configure MapReduce Task Tracker In {HADOOP_HOME}/conf/mapred-site.xml

```
<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
<description>The host and port that the MapReduce job tracker runs
at. If "local", then jobs are run in-process as a single map and reduce task.
</description>
</property>
```

(7) Configure Replication Number in {HADOOP_HOME}/conf/hdfs-site.xml

Hadoop lets user customize how many replications of a data block needed to be stored for the purpose of fault tolerance. Normally, it is configured to 3 in the cluster environment. Here we use 1 for the stand alone configuration.

```
<property>
<name>dfs.replication</name>
<value>1</value>
<description>Default block replication.
The actual number of replications can be specified when the file is created.
The default is used if replication is not specified in create time.
</description>
</property>
```

(8) Format The File System & Start

{HADOOP_HOME} /$ sudo bin/hadoop namenode -format
{HADOOP_HOME} /$ sudo bin/start-all.sh

(9) Run Examples

Copy the input files into the distributed file system:
```
$bin/hadoop fs -put conf input
```
Run some of the examples provided:
```
$ bin/hadoop jar hadoop-examples-*.jar grep input output 'dfs[a-
z.]+'
```
Examine the output files
Copy the output files from the distributed file system to the local file system and examine them:
```
$ bin/hadoop fs -get output output
$cat output/*
```